

Package equivalence in complex software network

Tomislav Slijepčević
University of Ljubljana
Faculty of Computer and Information Science
Večna pot 113, SI-1000 Ljubljana, Slovenia
ts2287@student.uni-lj.si

ABSTRACT

The public package registry *npm* is one of the biggest software registry. With its 216 911 software packages, it forms a big network of software dependencies. In this paper we evaluate various methods for finding similar packages in the *npm* network, using only the structure of the graph. Namely, we want to find a way of categorizing similar packages, which would be useful for recommendation systems. This size enables us to compute meaningful results, as it softened the particularities of the graph. *Npm* is also quite famous as it is the default package repository of *Node.js*. We believe that it will make our results interesting for more people than a less used package repository. This makes it a good subject of analysis of software networks.

Categories and Subject Descriptors

D.2.8 [Software Engineering]: Metrics—*complexity measures, performance measures, software science*

General Terms

Theory, algorithms, experimentation.

Keywords

Software systems, Software engineering, Software networks, Network analysis.

1. INTRODUCTION

The major interest of the network science is the analysis of the structural and statistical properties of complex networks. By looking at the patterns of connectivity in a network, a "role" can be assigned to each node [1, 2]. Researches have discovered that there is a close relation between the structure of a network and the roles, while they analyzed various networks that regards the life science, ecology, information, social sciences and economics [3, 4, 5, 6]. That is why it is important to understand the structure, i.e., the topology, of a network to find the roles and to understand the dynamics of a network in that matter [7].

The node gets a role based on how it interacts with others. For the role assignment there have been developed two basic concepts: structural [8] and regular equivalence [9]. When two nodes are either structurally or regularly equivalent, then they have the same role in a network. For two nodes to be structurally equivalent, they have to share the same neighbors. The regular equivalence is a looser criterion than the structural equivalence; it does not compare

the neighborhoods of the two nodes, but whether the two nodes are connected in the same way to the others. It holds that if two nodes are structurally equivalent, then they are also regularly equivalent, but not vice versa. The grouping of nodes that are related by either of equivalence relations forms equivalence classes, which represent the roles of nodes in a network.

In this paper we examine the complex software network to see if nodes can be naturally grouped into equivalence classes just as in other networks. The analyzed network consists of nodes that represent software packages and edges that indicate dependencies among packages. The data was taken from the biggest public software registry *npm* [10] (<https://npmjs.com>). We believe that applying the relation of structural equivalency on the software network would group the software packages into equivalence classes that represent different types of software, e.g. a class of packages that is related to web, graphics or other. Reasoning behind this is that packages that are related by type share many neighbors and would therefore be assigned to the same equivalence class. By applying the other equivalence relation, which is the regular equivalence, we assume that the packages will be grouped into the following three equivalence classes: a class of core libraries, a class of the popular frameworks, and a class of packages that are supplementary to the popular frameworks, i.e. packages that serve as an add-on to one of the popular framework. Packages within listed classes connects to others in the same way and that is why we predict the formation of these classes. For a class of the core libraries it is understandable that they do not have dependencies, as there is not any package to depend upon at start, but they do have a lot of dependents. For the packages within the second class, the popular frameworks, it reasonable that they have a lot of dependencies and a lot of dependents. The packages in the last class, which supplement the popular frameworks, should have a few dependencies and probably none dependents.

Thus it should be possible to find similar packages using only the structure of the network. This could lead to amelioration of package recommendations based on equivalence relations and also to automatic labeling/categorizing of a new package based just on the listed dependencies. This system could help to homogenize keywords in software registry and thus improve the referencing of the packages and simplify the search for packages.

The rest of the article is structured as follows. In Section 2 we present related work and in Section 3 we give a formal introduction to node equivalence. Empirical evaluation with discussion is done in Section 4 and conclusions in Section 5.

2. RELATED WORK

The modern science of network is particularly interested in decomposing nodes of large networks into independent groups called "communities" [11]. As a community, you think of a group of nodes that is internally densely connected but sparsely connected externally [7, 2, 11, 3]. The nodes within community are similar to each other and dissimilar to the rest of the nodes in a network. Researchers showed there is actually community structures in real-world networks [12, 11, 13]. However, the definition of community is not universally accepted and for that matter we have multiple definitions. Community structure emphasizes cohesive groups of nodes and the absence of dependencies between the groups, but this does not say anything about the roles in a network. The concept of roles in networks is much wider than the concept of community. Prerequisite for the analysis of roles in a network is a community structure. After that, you examine how discovered communities are interdependent, which translates to different roles in a network.

One of predominant technique for deriving structure of a network is blockmodeling [8]. It is considered as mapping nodes and edges onto their images in a reduced graph. A node ("block") in reduced graph represents the nodes from original graph that were mapped to the same image. This type of mapping is called semigroup homomorphism [9]. Homomorphisms are mappings that preserve the structure of a graph in a way that nodes of a graph are mapped into nodes in an image of the graph, and each edge in an image of the graph is connected to the same nodes as in the original graph. The semigroup is an algebraic structure consisting of a set with an associative binary operation. We can look at the nodes from original graph as a semigroup. Consequently, the nodes in reduced graph represent the result of applying the semigroup operation to the ordered pair (x, y) , which in our case are the edges in the original graph. If we take equivalence relation as the operation within semigroup, then we get equivalence classes as the nodes in the reduced graph. The plot of adjacency matrix of the new graph reveals blocks in diagonal, which are equivalence classes, and dependencies between them.

There exists a framework for blockmodeling classes within complex networks [7]. Authors derive a measure to find the best fitting image graph (reduced graph) and present a criterion to avoid overfitting. Image graph is termed as the role model. They did not demand exact mapping of every single node to the role model, but that the network as a whole fit as well as possible to the role model. The perfect fit would correspond to regular and structural equivalence. When tackling a new network, they assumed a given image graph and assignment of roles to nodes. They derived a quality function as an objective measure of fit between the image and the network under this assignment of roles. The assignment of roles which maximizes their quality function is considered as the best one to describe the connection structure of the original network. The concepts of modularity [14] and structural equivalence are found as special cases

of this approach. The modularity measure is commonly used in community detection algorithms. It measures the quality of a network division into communities. The proposed method is applicable to both two-mode and one-mode data, directed and undirected, as well as weighted networks. It is non-parametric and computationally efficient. In the same paper authors applied method to the world trade network and analyzed the roles individual countries play in the global economy.

Another research dealt with analysis of patterns of role-to-role connections, but with their own definition of roles [15]. Principle was the same; to group the nodes into roles, according to their pattern of intra- and intergroup connections. They analyzed four different types of real-world networks; metabolic networks, protein interactions, global and regional air transportation networks, and the Internet at the level of autonomous systems. To determine and quantify the modular structure of these networks, they use simulated annealing. The optimal partitions of the network into groups was found with the use of the modularity measure. By comparing modular structure of each network with the randomization of the same network, they found out that all observed networks have a significant modular structure. That is reasonable as groups in biological networks corresponds to functional units and in air transportation groups corresponds to geo-political units. The role of each node was determined according to two properties: the relative within-group degree z , which measures the node's degree of connectedness with the nodes within the same group, and the participation coefficient P , which measures node's degree of connectedness with other groups. When classifying nodes into roles, they initially measured within-group degree of all nodes, and divided them into hubs and non-hubs based on high and low within-module degree, respectively. Then they looked at the participation coefficient to further subdivide hubs into different types of hubs. Further subdivision was also done to non-hubs. Reasoning behind this particular definition of the roles is given in [13].

3. METHODS

A graph is an ordered pair $G = \langle P, R \rangle$, where P is a finite set of points and R is a relation on P , i.e., a subset of the ordered pairs of points in $P \times P$. An equivalence \equiv on P is a relation such that for all $a, b, c \in P$, it has a property of reflexivity ($a \equiv a$), symmetry ($a \equiv b \implies b \equiv a$) and transitivity ($a \equiv b \wedge b \equiv c \implies a \equiv c$) [9]. Grouping of points based on \equiv forms equivalence classes. Similarity in network analysis occurs when two nodes fall in the same equivalence class. There are two fundamental approaches for constructing measures of network similarity: structural equivalence [8] and regular equivalence [16]. There exists a hierarchy of these two equivalence concepts. Any structural equivalence is also regular equivalence, but not all regular equivalences are necessarily structural [17].

Structurally equivalent nodes of a network must share the same neighbors. Formally is defined as follows [9]: if $G = (P, R)$ and \equiv is an equivalence relation on P , then \equiv is structural equivalence if and only if for all $a, b, c \in P$ such that $a \neq c \neq b$, $a \equiv b$ implies: (i) $aRb \leftrightarrow bRa$, (ii) $aRc \leftrightarrow bRc$; and (iii) $cRa \leftrightarrow cRb$. The undirected graph in Figure 1 visually represents the concept, where nodes colored in red

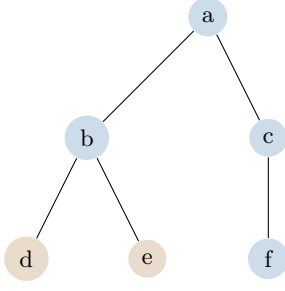


Figure 1: Structurally equivalent nodes are marked red.

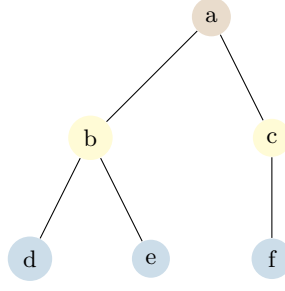


Figure 2: Regularly equivalent nodes are those sharing the same color.

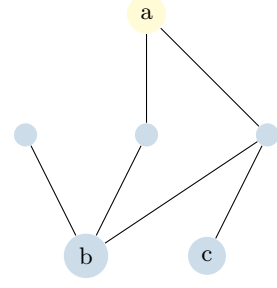


Figure 3: Problem of basic similarity measure for structural equivalence which just counts the number of common neighbors.

are the ones that are structurally equivalent.

Nodes that are regularly equivalent do not necessarily share neighbors, but have neighbors who are themselves similar. Formally is defined as follows [9]: if $G = (P, R)$ and \equiv is an equivalence relation on P then \equiv is a regular equivalence if and only if for all $a, b, c \in P$, $a \equiv b$ implies: (i) $aRc \implies \exists d \in P, bRd \wedge d \equiv c$; and (ii) $cRa \implies \exists d \in P, dRb \wedge d \equiv c$. This concept is visually depicted on the same undirected graph in Figure 2, where regularly equivalent nodes are colored with the same color.

Measuring structural equivalence of nodes

To find structurally equivalent nodes, one must compare each node with the rest and check which pairs of nodes have matching neighborhoods. This takes $\mathcal{O}(n^2)$ steps, as one is doing a pairwise comparison of n nodes in the network. Most of the comparisons are needless as not every node is connected to everyone else. We should only compare neighborhoods of connected nodes, and that is achieved if we traverse through the edges of a network, which takes $\mathcal{O}(m)$ steps, where m is the number of edges in the network. We consider edges (x, y) one at a time and compare neighborhoods of nodes x and y . We do not follow actual definition of structural equivalence, which gives binary answer (if neighborhoods match or not), but instead we measure to what extent they do match. The basic similarity measure just checks the size of the intersection of the two neighborhoods, i.e., the number of the common neighbors of two nodes [8]. The problem with this measure is that it is advantageously for nodes with bigger neighborhoods, which would therefore also have more common neighbors with the others. The example in Figure 3 shows the problem. By applying measure to pairs (b, a) and (c, a) , the measure would return two and one common neighbor, respectively, and would mean that node b is more structurally equivalent to node a than node c to node a . This is not true, as node b has additional neighbor, uncommon to node a . Problem can be solved by using the Salton cosine [18] similarity, which normalizes results with square root of degrees of connected nodes. If A is adjacency matrix, where A_{ij} indicates the number of neighbors between node i and j , and Γ_i is a set of neighbors of node i , and k_i is the degree of node i , then cosine similarity σ_{ij} is defined as:

$$\sigma_{ij} = \cos \theta_{ij} = \frac{\sum_k A_{ik} A_{jk}}{\sqrt{\sum_k A_{ik}^2} \sqrt{\sum_k A_{jk}^2}} = \frac{|\Gamma_i \cap \Gamma_j|}{\sqrt{k_i k_j}}$$

Before measuring structural equivalence of nodes in the network *npm*, we added additional edges. In a network of package dependencies such as *npm*, a package depends not only on those packages that it points to but also on those packages that its dependencies point to. When you install a package from the *npm*, the installer first installs dependencies of chosen package and then the package itself. But before dependencies are installed, it has to install their dependencies, and so on. So when you install a package, you get all its descendants. In that sense, a package depends on all descendants, therefore we added additional directed edges between a package and all its descendants. After that we proceeded with measuring structural equivalence of nodes. For two nodes to even be considered structurally equivalent they have to share at least one common neighbor. In our case two packages must depend to at least one same package. This is achieved if we pick each node and measure structural equivalence between pairs of its predecessors. This way a pair of predecessors will have at least picked node in common. When measuring structural equivalence of two nodes we first check if their neighbors overlap. If they do, then we mark those nodes as identical, otherwise we measure their cosine similarity. Afterwards we remove any edges that connects examined pairs, because they no longer need to be compared. We keep picking nodes and measuring similarity of its predecessors until we visited all nodes and all edges are removed. From all measured pairs we build a new undirected weighted network, where a pair of connected nodes is one of a measured pair, and each edge has a weight that is equal to a cosine similarity of connected nodes. Subsequently we contract nodes that we found to be identical, i.e., have identical neighbors. From the new network we extract similarity matrix, which we then give to a clustering method to find clusters of structurally equivalent nodes.

Measuring structural equivalence of nodes

For finding regularly equivalent nodes, we used the implementation of CATREGE [19] algorithm in R programming language [20]. It takes a graph and it assess how much are nodes regularly equivalent. For start, all nodes have the same role, therefore all are regularly equivalent. Then

those nodes within the same role but different combination of neighbor types are re-allocated to different roles. Initially neighbor type is the pattern of in- and out-connections. This procedure is then iterated until all nodes within each role have same combination of neighbor types. The distance between nodes in this case is the inverse of the number of iterative refinements of the initial role required to allocate the nodes to regularly equivalent roles. The distance of 0 indicates nodes which belong to the same role. The algorithm gives results in a form of similarity/dissimilarity matrix, which can then be used with a clustering method. The algorithm has a cubic complexity, therefore it is applicable only on a smaller graphs that consist of no more than thousand nodes.

Clustering with a distance matrix

We used k-medoids clustering algorithm [21], which is similar to k-means algorithm. Both algorithms are partitioning technique of clustering that clusters the data set of n objects into k clusters known a priori. They can take matrix of distances between points and partition points into clusters by minimizing the distance between a point in a cluster and a point designated as the center of that cluster (medoid). In contrast to the k-means algorithm, k-medoids works with an arbitrary distance between points instead of just euclidean distance. The most common realization of k-medoid clustering is the Partitioning Around Medoids (PAM). For start, PAM randomly selects k points as medoids. Then it associates each point to closest medoid and starts swapping medoids with non-medoids until cost of configuration stops decreasing. We used implementation of PAM in R programming language [22].

We applied PAM clustering on the similarity matrix of structural equivalence and dissimilarity matrix of regular equivalence. Beforehand, we transformed the similarity matrix of structural equivalence to dissimilarity matrix by subtracting 1 with cosine similarities. In this way, we got a cosine dissimilarity matrix. The matrices of both equivalence contain pairwise dissimilarity of all nodes in the network, therefore they take $O(n^2)$ space, where n is the number of nodes. Consequently, we cannot work with the whole *npm* network due to its big size. Therefore, we were obliged to sample the network and work with samples. We used simple random walk algorithm [23] for sampling. We do not know how many groups of structurally equivalent nodes are there, therefore we need to run PAM algorithm multiple times and set different k each time. Because of this time constrain, we sampled 1000 nodes. For finding groups of regularly equivalent nodes we sampled only 500 nodes due to cubic time complexity of CATREGE algorithm.

For determining k we used the silhouette method [24]. For each node i , let $a(i)$ be the average dissimilarity of i with all other nodes within the same cluster, and let $b(i)$ be the lowest average dissimilarity of i to any other cluster, of which i is not a member. The smaller the value $a(i)$, the better is a node assigned to the cluster. The cluster with this lowest average dissimilarity $b(i)$ is the next best cluster for the point i . Silhouette is defined as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} = \begin{cases} 1 - \frac{a(i)}{b(i)}, & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1, & \text{if } a(i) > b(i) \end{cases}$$

We are interested when $a(i)$ is small, which means node i is well matched to its cluster, and when $b(i)$ is bigger, which means node i is badly fit to neighboring cluster. Thus $s(i)$ will be close to one. If it equals to zero, then node fits to two clusters equally well, and if it is negative, then a node is more fit to neighboring cluster. The average $s(i)$ over all nodes of a cluster is a measure of how tightly grouped all the nodes in the cluster are. Thus it is appropriately measure of how the nodes has been clustered and can used to determine the number of clusters within a network.

4. RESULTS

Regular equivalence classes

For regular equivalence we sampled the *npm* network 20 times by randomly visiting 500 nodes. Within samples the mean number of edges is 1350 (std. dev. is 158). Each sample was clustered with PAM and $k \in [2, 20]$. The average silhouette value for each sample and each k is shown in Figure 4. The optimal number of clusters is ≈ 4 , which corresponds to the number of regular equivalence classes. The average silhouette value over all nodes is ≈ 0.6 , therefore nodes fit quite well to assigned clusters and regular classes. The silhouette plots of the first 3 samples is shown in Figure 6, where we can see that average silhouette value of nodes within the same cluster is more than 0.5, which is the another sign of the good clustering. Another kind of plots, called cluster plots, of the first 3 samples is shown in Figure 7. These plots show clustered nodes in the first 2 principal components, which capture $\approx 80\%$ of variance of dissimilarity matrix. We can see that nodes can actually be clustered in 3 clusters, and thus 2 or more clusters can be regarded as the same regular equivalence class. This is also nicely shown in level plots (block modeling) in Figure 8, where there are 3 prominent blocks which represent 3 regular equivalence classes. After examining the nodes in clusters, we found out that a cluster have either nodes that have only in-connections, or out-connection, or both in- and out-connections. There is a single cluster of nodes with only in-connections and a single cluster of nodes with only out-connections per sample. The rest clusters have nodes with both connections. The reason the nodes are not clustered in a single cluster is because they can be then further nicely divided by the scale of in- and/or out-connections. Our hypothesis states that there are following roles in the software networks: a group of core package with only in-connections, a group of popular packages with both connections, and a group of packages with only out-connections that supplement popular packages. A cluster of nodes with only in-connections represents a group of core packages, a cluster of nodes with both connections where there are more out- than in-connections represents a group of popular packages, and the rest clusters represent supplementary packages.

Structural equivalence classes

For structural equivalence we sampled the *npm* network 20 times by randomly visiting 1000 nodes. Within samples the

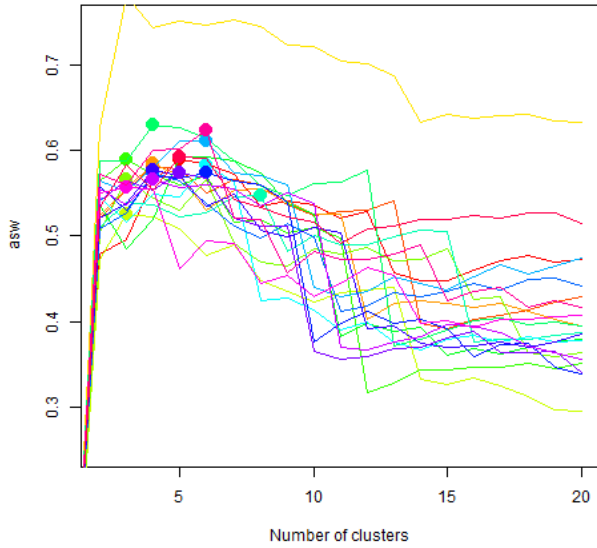


Figure 4: Determining the number of regular equivalence classes (clusters) in the *npm* network with average silhouette value.

mean number of edges is 8185 (std. dev. is 1711), the mean number of structurally equivalent node is 761 (std. dev. is 40), the mean number of edges between structurally equivalent nodes is 30772 (std. dev. is 13983), and the mean number of identical nodes is 270 (std. dev. is 40). Each sample was clustered with PAM and $k \in [1, 30]$. The average silhouette value for each sample and each k is shown in Figure 5. The optimal number of clusters cannot be determined because it differs from sample to sample, and therefore we cannot assess how many structural equivalence classes are there in the *npm* network. Although the average silhouette value of the best clustering per sample is small, we can see nice grouping of nodes into blocks in the level plots in Figure 9. These groups within the *npm* network could theoretically correspond to different sub-ecosystems, i.e., different types of software.

5. CONCLUSIONS

Nodes of the *npm* network were grouped by two equivalence relation: structural and regular equivalence. This was achieved by first measuring to what extent are pairs of nodes structurally/regularly similar/dissimilar. From pairwise similarities/dissimilarities we constructed dissimilarity matrix, which was then given to PAM clustering algorithm - the implementation of k-medoids. Because we had to work with matrix that takes quadratic space depending on a number of node in a network, we were obliged to work with a subgraphs. Due to this and other additional constraints, we could process at most thousand nodes per sample. Nonetheless, we successfully showed that regular equivalence relation when applied to the network indeed groups nodes into roles that we were seeking for. By applying structural equivalence to the network we found many different groups that could theoretically correspond to different types of software.

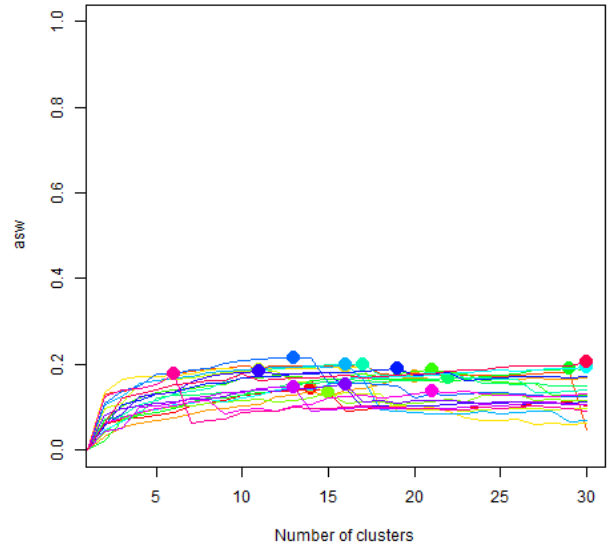


Figure 5: Determining the number of structural equivalence classes (clusters) in the *npm* network with average silhouette value.

6. REFERENCES

- [1] S. P. Borgatti and M. G. Everett, "Notions of position in social network analysis," *Sociological methodology*, vol. 22, no. 1, pp. 1–35, 1992.
- [2] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*, vol. 8. Cambridge university press, 1994.
- [3] M. E. Newman, "The structure and function of complex networks," *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.
- [4] A.-L. Barabási and Z. N. Oltvai, "Network biology: understanding the cell's functional organization," *Nature reviews genetics*, vol. 5, no. 2, pp. 101–113, 2004.
- [5] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, no. 6804, pp. 651–654, 2000.
- [6] H. Jeong, S. P. Mason, A.-L. Barabási, and Z. N. Oltvai, "Lethality and centrality in protein networks," *Nature*, vol. 411, no. 6833, pp. 41–42, 2001.
- [7] J. Reichardt and D. R. White, "Role models for complex networks," *The European Physical Journal B*, vol. 60, no. 2, pp. 217–224, 2007.
- [8] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *The Journal of Mathematical Sociology*, vol. 1, pp. 49–80, 1971.
- [9] "Graph and semigroup homomorphisms on networks of relations," vol. 5, no. 2.
- [10] <http://www.modulecounts.com/>.
- [11] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

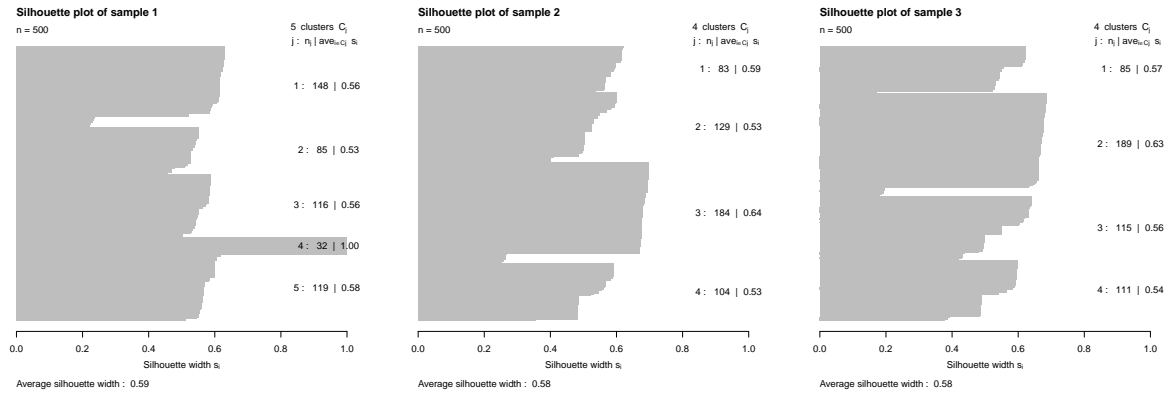


Figure 6: Silhouette plots of the first 3 samples for determining the number of regular equivalence classes in *npm* network.

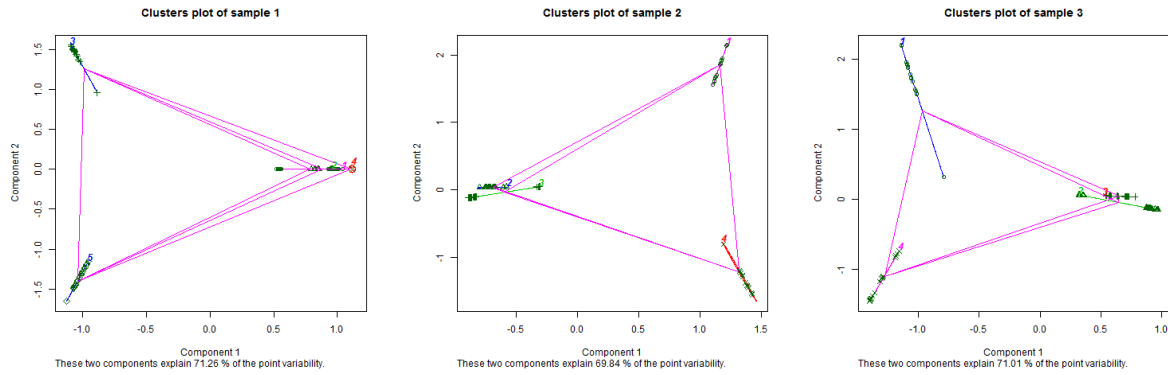


Figure 7: Clusters plots of the first 3 samples for determining the number of regular equivalence classes in *npm* network.

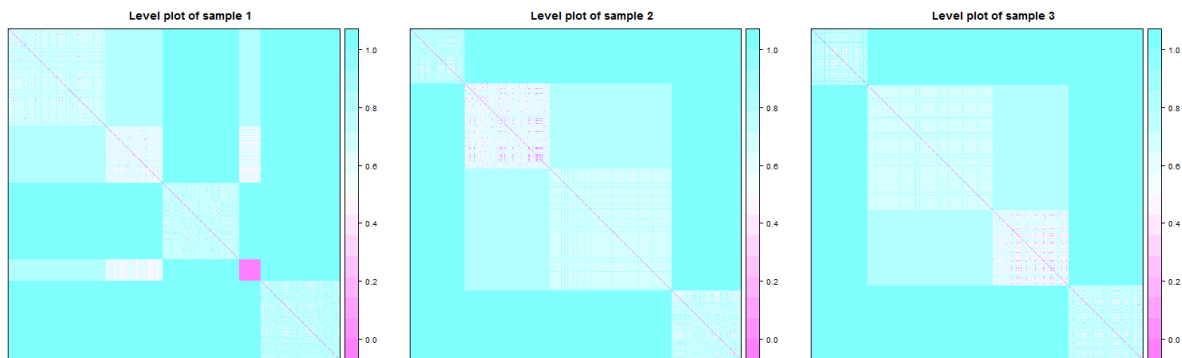


Figure 8: Level plots (block modeling) of the first 3 samples for determining the number of regular equivalence classes in *npm* network.

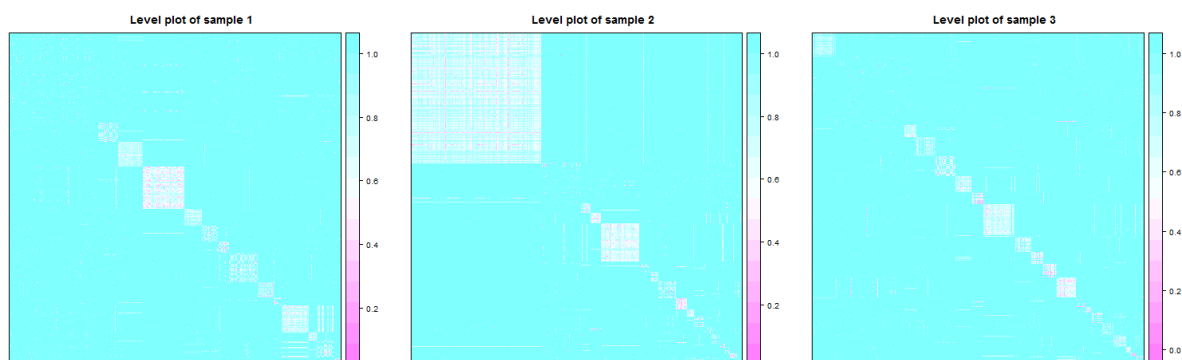


Figure 9: Level plots (block modeling) of the first 3 samples for determining the number of structural equivalence classes in *npm* network.

- [12] G. W. Flake, S. Lawrence, and C. L. Giles, "Efficient identification of web communities," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 150–160, ACM, 2000.
- [13] R. Guimera and L. A. N. Amaral, "Functional cartography of complex metabolic networks," *Nature*, vol. 433, no. 7028, pp. 895–900, 2005.
- [14] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.
- [15] R. Guimerà, M. Sales-Pardo, and L. A. N. Amaral, "Classes of complex networks defined by role-to-role connectivity profiles," *Nature physics*, vol. 3, 2007.
- [16] M. Newman, *Networks: an introduction*. Oxford University Press, 2010.
- [17] R. A. Hanneman and M. Riddle, "Introduction to social network methods," 2005.
- [18] G. Salton and M. J. McGill, "Introduction to modern information retrieval," *McGraw-Hill*, 1983.
- [19] S. P. Borgatti and M. G. Everett, "Two algorithms for regular equivalence," *Social networks*, vol. 15, pp. 361–376, 1993.
- [20] C. T. Butts, *sna: Tools for Social Network Analysis*, 2014. R package version 2.3-2.
- [21] L. Kaufman and P. Rousseeuw, *Clustering by means of medoids*. North-Holland, 1987.
- [22] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik, *cluster: Cluster Analysis Basics and Extensions*, 2015. R package version 2.0.3 — For new features, see the 'Changelog' file (in the package source).
- [23] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 631–636, ACM, 2006.
- [24] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.